Information Systems 45 (2014) 61–68

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Approximate nearest neighbor algorithm based on navigable small world graphs

Yury Malkov[a,b,*], Alexander Ponomarenko[b,c], Andrey Logvinov[b], Vladimir Krylov[b,d]

[a] The Institute of Applied Physics of the Russian Academy of Sciences, 46 Ul'yanov Street, 603950 Nizhny Novgorod, Russia
[b] MERA Labs LLC, 13, Delovaya St., Nizhny Novgorod 603163, Russia
[c] National Research University Higher School of Economics, Laboratory of Algorithms and Technologies for Network Analysis, 136 Rodionova, Nizhny Novgorod 603093, Russia
[d] Nizhny Novgorod State Technical University, Nizhny Novgorod, Russia

ARTICLE INFO

Available online 4 November 2013

Keywords:
Similarity search
k-Nearest neighbor
Approximate nearest neighbor
Navigable small world
Distributed data structure

ABSTRACT

We propose a novel approach to solving the approximate *k*-nearest neighbor search problem in metric spaces. The search structure is based on a navigable small world graph with vertices corresponding to the stored elements, edges to links between them, and a variation of greedy algorithm for searching. The navigable small world is created simply by keeping old Delaunay graph approximation links produced at the start of construction. The approach is very universal, defined in terms of arbitrary metric spaces and at the same time it is very simple. The algorithm handles insertions in the same way as queries: by finding approximate neighbors for the inserted element and connecting it to them. Both search and insertion can be done in parallel requiring only local information from the structure. The structure can be made distributed. The accuracy of the probabilistic k-nearest neighbor queries can be adjusted without rebuilding the structure.

The performed simulation for data in the Euclidean spaces shows that the structure built using the proposed algorithm has small world navigation properties with $\log^2(n)$ insertion and search complexity at fixed accuracy, and performs well at high dimensionality. Simulation on a CoPHiR dataset revealed its high efficiency in case of large datasets (more than an order of magnitude less metric computations at fixed recall) compared to permutation indexes. Only 0.03% of the 10 million 208-dimensional vector dataset is needed to be evaluated to achieve 0.999 recall (virtually exact search). For recall 0.93 processing speed 2800 queries/s can be achieved on a dual Intel X5675 Xenon server node with Java implementation.

1. Introduction

The scalability of any software system is limited by the scalability of its data structures. Massively distributed systems like BitTorrent or Skype are based on distributed hash tables. While the latter structures have good scalability, their search functionality is limited to the exact matching. This limitation arises because small changes in an element value lead to large and chaotic changes in the hash value, making the hash-based approach inapplicable to the range search and the similarity search problems.

However, there are many applications (such as pattern recognition and classification [1], content-based image retrieval [2], machine learning [3], recommendation systems [4], searching similar DNA sequence [5], semantic

* Corresponding author at: The Institute of Applied Physics of the Russian Academy of Sciences, 46 Ul'yanov Street, 603950 Nizhny Novgorod, Russia.
E-mail address: yurymalkov@mail.ru (Y. Malkov).

0306-4379/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2013.10.006

document retrieval [6]) that require the similarity search rather than just exact matching. The k-nearest neighbor search (k-NNS) problem is a mathematical formalization for similarity search. It is defined as follows: we need to find the set of k closest objects $P \subseteq X$ from a finite set of objects $X \subseteq \mathcal{D}$ to a given query $q \in \mathcal{D}$, where $\mathcal{D}$ is the set of all possible objects (the data domain). Closeness or proximity of two objects $o', o'' \in \mathcal{D}$ is defined as a distance function $\delta(o', o'')$.

A naïve solution for the k-NNS problem is to calculate the distance function δ between q and every element from X. This leads to linear search time complexity, which is much worse than the scalability of structures for exact match search, and makes the naïve version of k-NNS almost impossible to use for large size datasets.

We suggest a solution for the nearest neighbor search problem: a data structure represented by a graph $G(V, E)$, where every object o_i from X is uniquely associated with a vertex v_i from V. Searching for the closest elements to the query q from the data set X takes the form of searching for a vertices in the graph G.

This gives an opportunity for building decentralized similarity search oriented storage systems where physical data location does not depend on the content because every data object can be placed on an arbitrary physical machine and can be connected with others by links like in p2p systems.

One of the basic vertex search algorithms in graphs with metric objects is the greedy search algorithm. It has a simple implementation and can be initiated from any vertex. In order for the algorithm to work correctly (always return precise results), the network must contain the Delaunay graph as its subgraph, which is dual to the Voronoi tessellation [7]. However, there are major drawbacks associated with the Delaunay graph: it requires some knowledge of metric space internal structure [8] and it suffers from the curse of dimensionality [7]. Moreover, for the applications described above, the precise exactness of the search is not required. So the problem of finding the exact nearest neighbors can be substituted by the approximate nearest neighbor search, and thus we do not need to support the whole/exact Delaunay graph.

Graphs with logarithmic scalability of the greedy search algorithm are called navigable small world graphs, they are well known in Euclidean spaces [9]. Note that the small world models (not *navigable* small world) like [10] do not have this feature. Even though there are short paths in the graph, the greedy algorithm do no tend to find them, in the end having a power law search complexity. Solutions for constructing a navigational small world graphs were proposed for general spaces but they are usually more complex, requiring sampling, iterations, rewiring etc. [11–14]. We show that the small world navigation property can be achieved with a much simpler technique even without prior knowledge of internal structure of a metric space (e.g. dimensionality or data density distribution).

In this paper we present a simple algorithm for the data structure construction based on a navigable small world network topology with a graph $G(V, E)$, which uses the greedy search algorithm for the approximate k-nearest neighbor search problem. The graph $G(V, E)$ contains an approximation of the Delaunay graph and has long-range links together with the small-world navigation property. The search algorithm we propose has the ability to choose the accuracy of search without modification of the structure. Presented algorithms do not use the coordinate representation and do not presume the properties of Euclidean spaces, because they are based only on comparing distances between the objects and the query, and therefore in principle are applicable to data from general metric (or even non-metric) spaces. Simulations revealed weak dimensionality dependence for Euclidean data.

2. Related work

Kd-tree [15] and quadra trees [16] were among the first works on the kNN problem. They perform well in 2–3 dimensions (search complexity is close to $O(\log n)$ in practice), but the analysis of the worst case for these structures [17] indicates $O(d^* N^{1-1/d})$ search complexity, where d is the dimensionality.

In Ref. [8] was proposed an exact-proximity search structure that uses the Delaunay graph with the greedy search algorithm. Authors showed the impossibility of finding the exact Delaunay graph in a general metric space, and to keep the search exact they resort to backtracking. Proposed data structure has construction time $O(n \log^2 n / \log \log n)$ and search time $O(n^{1-\Theta(1/\log \log n)})$ in high dimensions and $O(n^{\alpha})$, $(0 < \alpha < 1)$in low dimensions.

In general, currently there are no methods for effective exact NNS in high-dimensionality metric spaces. The reason behind this lies in the “curse” of dimensionality [18]. To avoid the curse of dimensionality while retaining the logarithmic cost on the number of elements, it was proposed to reduce the requirements for the kNN problem solution, making it approximate (Approximate kNN).

There are two commonly used definitions of the approximate neighbor search. One class of methods proposed to search with predefined accuracy ε (ε-NNS). It means that the distance between the query and any element in the result is no more than $1+\varepsilon$ times the distance from query to its true k-th nearest neighbor. Such methods have been described in [19–23]. Another class gives probability guarantee of finding true k closest point to the query [24–31], using “recall” (the fraction of true k nearest elements found).

Some structures [19–23] can be applied only to Euclidean space. Other methods [24–31] are applicable to the general metric space. More can be found in reviews [32,33].

Permutation indexes (PI) [25,34] is an efficient non-distributed algorithm suitable for general metric spaces. The idea behind PI is to represent each database object with the permutation of a set of references, called the permutants, sorted by distance to the object. The distance between objects is hinted by the distance between their respective permutations. PI is known to have high precision and recall even for datasets with high intrinsic dimensionality.

The work by Houle and Sakuma [26] features a probabilistic tree-like structure for the approximate nearest neighbor search in general metric spaces, based on selection of the nearest neighbors. The algorithm was simulated